Q1. Sort a list of students by roll number (ascending) using Comparable.

Create a Student class with fields: rollNo, name, and marks. Implement the Comparable interface to sort students by their roll numbers.

```
package Assesement_day9;
import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

class Student implements Comparable<Student> {

	int rollNo;

	String name;

	double marks;

	public Student(int rollNo, String name, double marks) {

		this.rollNo = rollNo;

		this.name = name;

		this.marks = marks;

	}

	public int compareTo(Student other) {

		return Integer.compare(this.rollNo, other.rollNo);

	}

	public String toString() {
```

```
        return "Roll No: " + rollNo + ", Name: " + name + ",
Marks: " + marks;
    }
}
public class Sort_list {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student(3, "Sanjana", 85.5));
        students.add(new Student(1, "Dhana", 90.0));
        students.add(new Student(2, "Sri", 78.0));
        students.add(new Student(4, "Penugonda", 92.0));

        System.out.println("Before sorting:");
        for (Student student : students) {
            System.out.println(student);
        }
        Collections.sort(students);
        System.out.println("\nAfter sorting:");
        for (Student student : students) {
            System.out.println(student);
        }
    }
```

}

Output:

Before sorting:

Roll No: 3, Name: Sanjana, Marks: 85.5

Roll No: 1, Name: Dhana, Marks: 90.0

Roll No: 2, Name: Sri, Marks: 78.0

Roll No: 4, Name: Penugonda, Marks: 92.0

After sorting:

Roll No: 1, Name: Dhana, Marks: 90.0

Roll No: 2, Name: Sri, Marks: 78.0

Roll No: 3, Name: Sanjana, Marks: 85.5

Roll No: 4, Name: Penugonda, Marks: 92.0]

Q2. Create a Product class and sort products by price using Comparable.

Implement Comparable<Product> and sort a list of products using Collections.sort().

```
package Assesement_day9;
import java.util.ArrayList;
import java.util.Collections;

class Product implements Comparable<Product> {
	String name;
	double price;
	public Product(String name, double price) {
		this.name = name;
		this.price = price;
	}
	public int compareTo(Product other) {
		return Double.compare(this.price, other.price);
	}
	public String toString() {
		return "Product: " + name + ", Price: " + price;
	}
}

public class Collection_sort {
	public static void main(String[] args) {
		ArrayList<Product> products = new ArrayList<>();
		products.add(new Product("Laptop", 50000));
		products.add(new Product("Mobile", 20000));
		products.add(new Product("Tablet", 30000));

		Collections.sort(products);
		for (Product product : products) {
			System.out.println(product);
		}
	}
```

```
}
```

Output:

```
Product: Mobile, Price: 20000.0
Product: Tablet, Price: 30000.0
Product: Laptop, Price: 50000.0
```

Q3. Create an Employee class and sort by name using Comparable.

Use the compareTo() method to sort alphabetically by employee names.

```
package Assesement_day9;
import java.util.Arrays;

class Employee implements Comparable<Employee> {
	String name;
	int id;
	public Employee(String name, int id) {
		this.name = name;
		this.id = id;
	}
	public int compareTo(Employee other) {
		return this.name.compareTo(other.name);
	}
	public String toString() {
```

```
        return "Employee: " + name + ", ID: " + id;
    }
}
public class Sorting {
    public static void main(String[] args) {
        Employee[] employees = {
            new Employee("Dhana", 101),
            new Employee("Sri", 102),
            new Employee("Sanjana", 103)
        };
        Arrays.sort(employees);
        for (Employee employee : employees) {
            System.out.println(employee);
        }
    }
}
```

Output:

```
Employee: Dhana, ID: 101
Employee: Sanjana, ID: 103
Employee:    Sri, ID: 102
```

Q4. Sort a list of Book objects by bookId in descending order using Comparable.

Hint: Override compareTo() to return the reverse order

```
package Assesement_day9;
import java.util.ArrayList;
import java.util.Collections;


class Book implements Comparable<Book> {
	int bookId;
	String title;


	public Book(int bookId, String title) {
		this.bookId = bookId;
		this.title = title;
	}
	public int compareTo(Book other) {
			return Integer.compare(other.bookId, this.bookId);
	}
	public String toString() {
		return "Book ID: " + bookId + ", Title: " + title;
	}
```

```
}
public class list_of_books {
	public static void main(String[] args) {
		ArrayList<Book> books = new ArrayList<>();
		books.add(new Book(101, "Book A"));
		books.add(new Book(103, "Book C"));
		books.add(new Book(102, "Book B"));

		Collections.sort(books);
		for (Book book : books) {
			System.out.println(book);
		}
	}
}
```

Output:

Book ID: 103, Title: Book C

Book ID: 102, Title: Book B

Book ID: 101, Title: Book A

Q5. Implement a program that sorts a list of custom objects using Comparable, and displays them before and after sorting

```
package Assesement_day9;
import java.util.ArrayList;
import java.util.Collections;


class Student implements Comparable<Student> {
	String name;
	int age;

	public Student(String name, int age) {
		this.name = name;
		this.age = age;
	}
	public int compareTo(Student other) {
		return this.name.compareTo(other.name);
	}
	public String toString() {
		return "Name: " + name + ", Age: " + age;
	}
}
```

```
public class Sorting_asc {
	public static void main(String[] args) {
		ArrayList<Student> students = new ArrayList<>();
		students.add(new Student("Sanjana", 20));
		students.add(new Student("Dhana", 22));
		students.add(new Student("Sri", 21));

		System.out.println("Before Sorting:");
		for (Student student : students) {
			System.out.println(student);
		}
		Collections.sort(students);
		System.out.println("\nAfter Sorting:");
		for (Student student : students) {
			System.out.println(student);
		}
	}
}
```

Output:

Before Sorting:

Name: Sanjana, Age: 20

Name: Dhana, Age: 22

Name: Sri, Age: 21

After Sorting:

Name: Dhana, Age: 22

Name: Sanjana, Age: 20

Name: Sri, Age: 21

Q1. Create and Write to a File

Write a Java program to create a file named student.txt and write 5 lines of student names using FileWriter.

```
package Assesement_day9;
import java.io.FileWriter;

import java.io.IOException;

public class File {

	public static void main(String[] args) {

		String[] studentNames = {"Sanjana", "Dhana", "Sri",
"Penugonda", "Prasanna"};


		try (FileWriter writer = new FileWriter("student.txt"))
{

			for (String name : studentNames) {

				writer.write(name + "\n");

			}
```

```
                System.out.println("Student names written to
student.txt");
            } catch (IOException e) {
                System.out.println("Error writing to file: " +
e.getMessage());
            }
        }
}
```

Output:

Sanjana

Dhana

Sri

Penugonda

Prasanna

Q2. Read from a File

Write a program to read the contents of student.txt and display them line by line using BufferedReader.

```
package Assesement_day9;
import java.io.BufferedReader;
import java.io.FileReader;
```

```
import java.io.IOException;


public class Read_file {
	public static void main(String[] args) {
		try (BufferedReader reader = new
BufferedReader(new FileReader("student.txt"))) {
			String line;
			while ((line = reader.readLine()) != null) {
				System.out.println(line);
			}
		} catch (IOException e) {
			System.out.println("Error reading file: " +
e.getMessage());
		}
	}
}
```

Output:

Sanjana

Dhana

Sri

Penugonda

Prasanna

Q3. Append Data to a File

Write a Java program to append a new student name to the existing student.txt file without overwriting existing data.

```
package Assessement_day9;
import java.io.FileWriter;
import java.io.IOException;

public class append_data {

public static void main(String[] args) {
try (FileWriter writer = new FileWriter("student.txt", true)) {
writer.write("Srinu\n");
System.out.println("New student name appended to file.");
} catch (IOException e) {
System.out.println("Error appending to file: " +
e.getMessage());
}
}

}
```

Output:

New student name appended to file.

Q5. Copy Contents from One File to Another

Write a program to read from source.txt and write the same content into destination.txt.

```
package Assessement_day9;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class copy_content {

public static void main(String[] args) {
try (BufferedReader reader = new BufferedReader(new
FileReader("date.txt"));
BufferedWriter writer = new BufferedWriter(new
FileWriter("destination.txt"))) {

String line;
System.out.println("Content of source.txt:");
while ((line = reader.readLine()) != null) {
System.out.println(line);
writer.write(line);
writer.newLine();
}
System.out.println("\nFile copied successfully. Content of
destination.txt:");
printFileContent("destination.txt");
```

```
} catch (IOException e) {
System.out.println("Error copying file: " + e.getMessage());
}
}

private static void printFileContent(String filename) {
try (BufferedReader reader = new BufferedReader(new
FileReader(filename))) {
String line;
while ((line = reader.readLine()) != null) {
System.out.println(line);
}
} catch (IOException e) {
System.out.println("Error reading file: " + e.getMessage());
}

}

}
```

Output:

Content of source.txt:
Hello to World!

File copied successfully. Content of destination.txt:

---

6. Reverse File Content

Write a program to read a file data.txt and create another file reversed.txt containing the lines in reverse order.

```
package Assessement_day9;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Reverse_file {

public static void main(String[] args) {
List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new
FileReader("data.txt"))) {
String line;
while ((line = reader.readLine()) != null) {
lines.add(line);
}
} catch (IOException e) {
System.out.println("Error reading file: " + e.getMessage());
return;
}

Collections.reverse(lines);
```

```
try (BufferedWriter writer = new BufferedWriter(new
FileWriter("reversed.txt"))) {
for (String line : lines) {
writer.write(line);
writer.newLine();
}
System.out.println("File content reversed successfully");
} catch (IOException e) {
System.out.println("Error writing to file: " + e.getMessage());
}

}

}
```

Output:
File content reversed successfully

---

7. Delete a File

Write a program to delete a file (given by file name) if it exists.

```
package Assessement_day9;
import java.io.File;
import java.util.Scanner;
public class Delete_file {
```

```
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the file name: ");
String fileName = scanner.nextLine();
scanner.close();

File file = new File(fileName);
if (file.exists() && file.isFile()) {
if (file.delete()) {
System.out.println("File deleted successfully.");
} else {
System.out.println("Failed to delete the file.");
}
} else {
System.out.println("File does not exist.");
}

}

}
```

Output:

```
Enter the file name: simple.txt
File deleted successfully.
```

---

8. Replace a Word in a File

Read content from story.txt, replace all occurrences of the word "Java" with "Python", and write the updated content to updated_story.txt

```
package Assessement_day9;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Replace_file {

public static void main(String[] args) {
try (BufferedReader reader = new BufferedReader(new
FileReader("hello.txt"));
BufferedWriter writer = new BufferedWriter(new
FileWriter("updated_story.txt"))) {

String line;
while ((line = reader.readLine()) != null) {
String updatedLine = line.replaceAll("Java", "Python");
writer.write(updatedLine);
writer.newLine();
}
System.out.println("Word replaced successfully. Updated
content written to updated_story.txt");

} catch (IOException e) {
System.out.println("Error reading or writing file: " +
e.getMessage());
}

}
```

}

Output:

Word replaced successfully. Updated content written to updated_story.txt